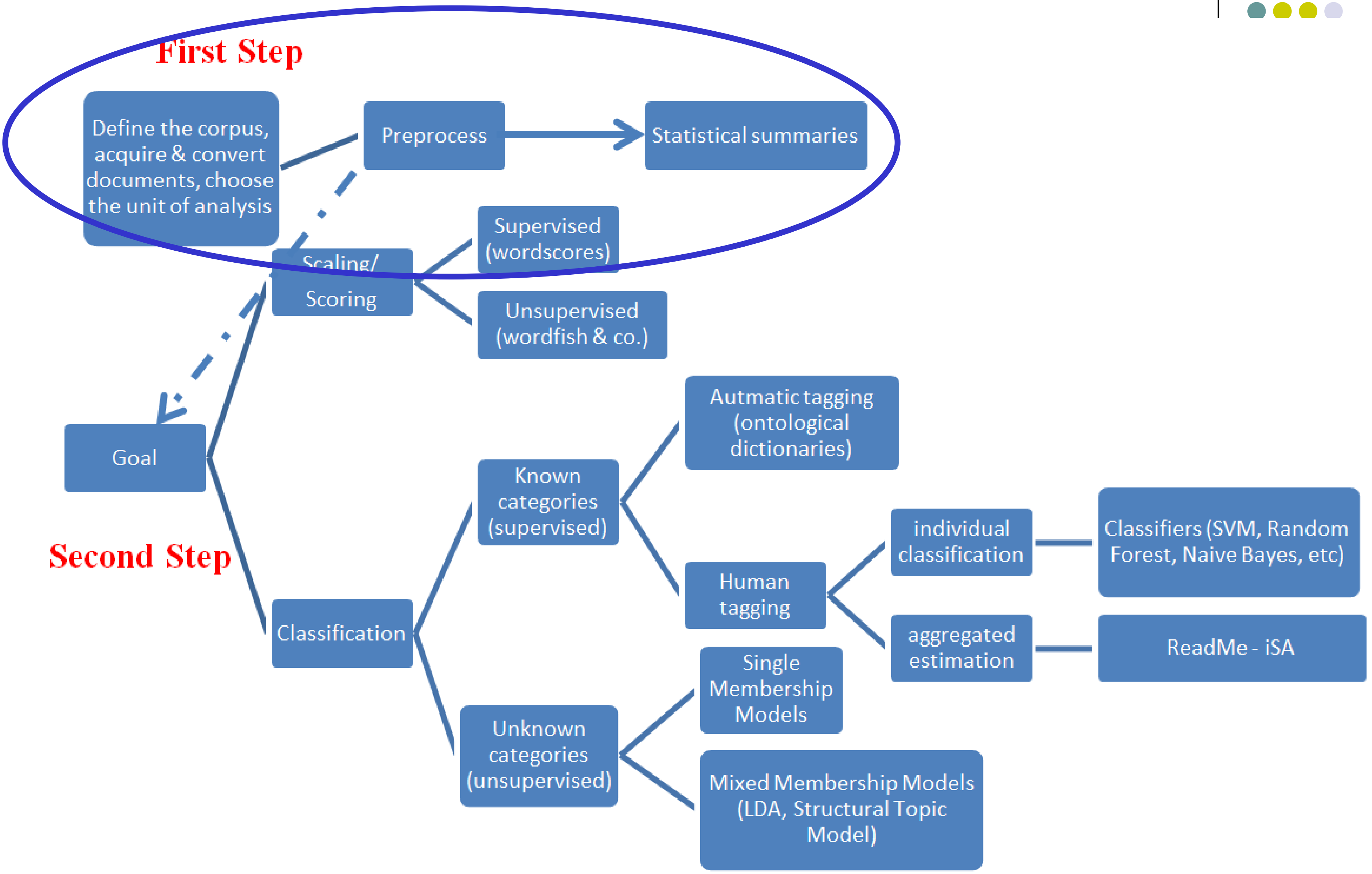
Big Data Analytics

Lecture 1 (second part)

How to prepare a text for analysis

Our Course Map


First Step
Define the corpus, acquire & convert documents, choose the unit of analysis
Preprocess
Statistical summaries
Scaling/ Scoring
Supervised (wordscores)
Unsupervised (wordfish & co.)
Goal
Second Step
Classification
Known categories (supervised)
Autmatic tagging (ontological dictionaries)
Human tagging
individual classification
Classifiers (SVM, Random Forest, Naive Bayes, etc)
aggregated estimation
ReadMe - iSA
Unknown categories (unsupervised)
Single Membership Models
Mixed Membership Models (LDA, Structural Topic Model)

The First Step: the preparation

Two stages:

1. **Defining the corpus** and the unit of analysis, acquiring the texts
2. **Preprocessing stage**: defining and refining features as well as converting textual features into a quantative matrix

Define the corpus

Jargon: we refer to *text* or *document* as the **unit of analysis** (it could apply to any unit of text: a tweet, a Facebook status, press briefing, sentence, paragraph)

We refer to the population of texts to be analyzed as the *corpus* and a collection of these as *corpora*

Define the corpus

A year of articles about the economy from The New York Times, for instance, could form a **corpus** for analysis, where the **unit** (text or document) of analysis is an article

A set of debates during (one of the many) votes on Brexit in the UK House of Commons could form another **corpus**, where the **unit** of analysis is a speech act (one intervention by a speaker on the floor of parliament)

Italia-language party election manifestos from 1948 to 2018 could form a **corpus**, where a **unit** of analysis is a manifesto

Define the corpus

Defining the corpus is not irrelevant at all!

As a researcher you need to ensure that the texts under examination are related to the **research question you are interest about** and have **theoretical consistency**

Acquire the texts

In our attempt to acquire our corpus, we want **to include** in the corpus all relevant texts (i.e., *minimize false negatives*) and **exclude** any irrelevant texts (i.e., *minimize false positives*)

For example, imagine that you want to retrieve your corpus from Twitter by using a list of keywords

Acquire the texts

In this case you want to generate a *list of keywords* expected to distinguish between tweets relevant to the topic you are interest about (say, *Donald Trump*) compared to irrelevant tweets

It is however critical that the analyst *pay attention* to selecting keywords that are both relevant to the population of interest (given the topic you care about) and representative of the population of interest (i.e., not being too narrow and selecting only the tweets pro or against Donald Trump via a biased list of keywords)

Acquire the texts

The burst of interest in automated content methods is mainly due to the proliferation of **easy-to-obtain** digital texts

Some of these texts are already available (for example, legislative speeches), other should be recollected by you

Later on we will discuss how to retrieve data from social media (i.e., Twitter, possibly YouTube and TikTok)

Convert the texts

The step of converting the texts into a common electronic format is a purely technical one, involving no research design decisions, but it can nonetheless poses one of the stickiest problems in text analysis (pdf as image…)

Preprocessing stage

But then…how to move from words to number? That is:

- *how a text can be transformed into digital data so that an algorithm can then treat it?*

Preprocessing stage

Introducing some terms…

Words as they occur in a text are commonly known as **tokens**, so that the text “*one two one two*” contains four tokens

Tokenization is the process of **splitting a text** into its constituent tokens

Tokenization usually happens by recognizing the delimiters between words, which in most languages takes the form of a space. In more technical language, inter-word delimiters are known as **whitespace**, and include additional machine characters such as newlines, tabs, and space variants

Most languages separate words by whitespace, but some major ones such as Chinese, Japanese, and Korean do not

Preprocessing stage

For example, Japanese sentence is only distinguished by commas and periods, and words are put in sequence without spaces in between. And so?

Tokenizing these languages requires a *set of rules* to recognize word boundaries, usually from a listing of common word endings

Preprocessing stage

私は、日本社会党を代表して、当面する内外の諸問題につき、佐藤総理大臣にその所見をただださんとするものであります。

↓ after tokenization

私_は_、_日本_社会党_を_代表_し_て_、_当面_する 内外_の_諸問題_に_つき_、_佐藤_総理_大臣_に_その_所見_を_ただださん_と_する_もの_で_あり_ます_。

Preprocessing stage

To introduce another term, **word types** refer to uniquely occurring words

So that the text "*one two one two*" contains four tokens, but only two word types, "one" and "two"

Preprocessing stage

For a **token** to become a **feature** of textual data (our basic unit of analysis), it typically undergoes transformation in a step often called “pre-processing”

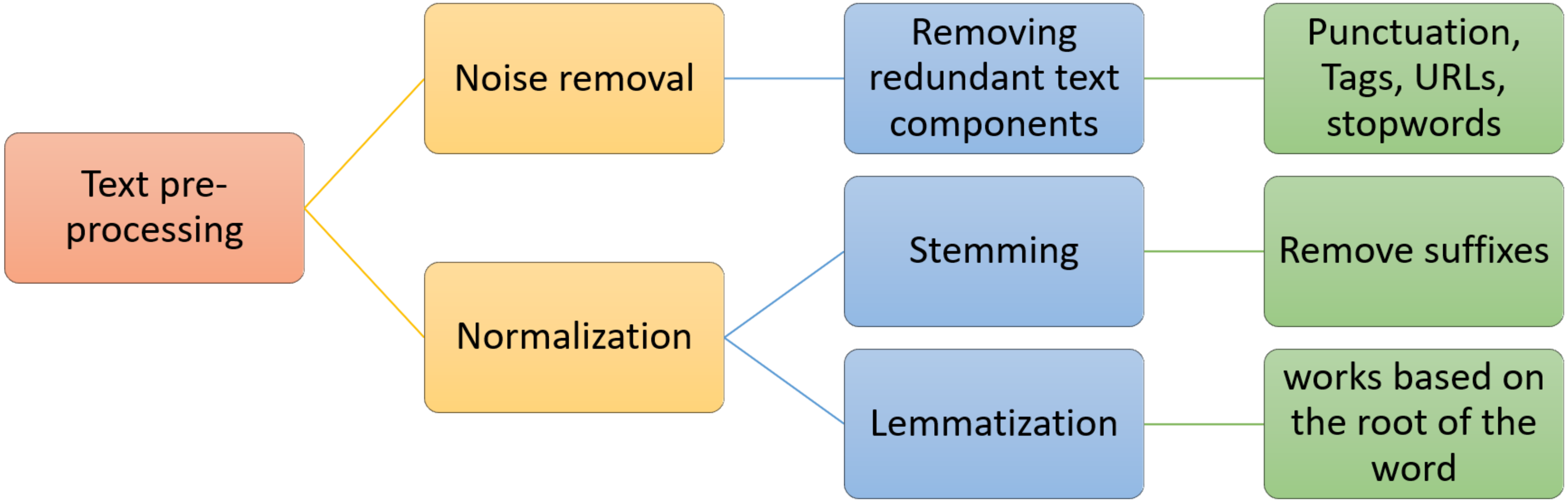
Why such transformation is needed? Cause language is **complex**! But not all of language’s complexity is necessary to effectively analyze texts (REMEMBER?)

We should **retain information** that will be used by the automated methods, **while discarding information** that will likely be unhelpful, ancillary, or too complex for use in a statistical model

In other words: there are many forms of “words”, and these typically undergo a process of selection and transformation before they become **features** of our textual dataset

Preprocessing stage

Text pre-processing can be divided into two broad categories—**noise removal & normalization**


Text pre-processing
Noise removal
Normalization
Removing redundant text components
Stemming
Lemmatization
Punctuation, Tags, URLs, stopwords
Remove suffixes
works based on the root of the word

Preprocessing stage

1. **Noise removal**: Data components that are redundant to the core text analytics can be considered as **noise**

Such as?!?

The First Step: the preparation

Stopwords! They include the large number of prepositions, pronouns, conjunctions etc. in sentences such as *the, is, at, which,* and *on* in English that occur in the greatest frequency in natural language texts

These words can be considered **unlikely** to contribute useful information for analysis, adding little specific political meaning to the text

However…

The First Step: the preparation

…the pronoun “**her”**, as Monroe, Quinn and Colaresi (2008) found, has a decidedly partisan orientation in debates on abortion in the U.S. Senate

For these reasons, when preparing textual data for analysis, always check the impact on your final results of eliminating or not stopwords…

The First Step: the preparation

We also typically discard:

- **Punctuation**
- **Capitalization**: we apply lower-casing, which treats words as equivalent regardless of how they were capitalised
- We can also decide to eliminate words through the use of **predefined lists of words to be ignored** (for example: tags, URLs, etc.) or based on **their relative infrequency** (words that appear only once or twice in the corpus are unlikely to be discriminating)

The First Step: the preparation

2. **Normalization**: Handling multiple occurrences / representations of the same word is called normalization

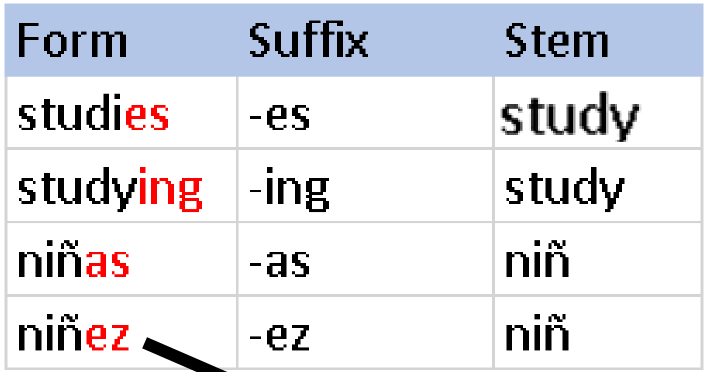
There are two types of normalization: **stemming** and **lemmatization**

The First Step: the preparation

Stemming normalizes text by reducing words to their stems, which is a cruder algorithmic means of equating a word with its canonical (dictionary) form, i.e., stemming treats **words as equivalent** when they differ only in their inflected forms

For example, the different words *taxes, tax, taxation, taxing, taxed, and taxable* are all converted to their word stem "**tax**"

The First Step: the preparation



Form	Suffix	Stem
studies	-es	study
studying	-ing	study
niñas	-as	niñ
niñez	-ez	niñ

$\longrightarrow$ **Stemming**

Girls

Childhood

By doing that, stemming reduce the total number of unique words in the data set

The First Step: the preparation

Lemmatization is a more advanced technique which works based on the **root of the word** taking into consideration the **morphological analysis of the words**

To do so, it is necessary to have detailed dictionaries which the algorithm can look through to link the form back to its lemma

The First Step: the preparation

Form	Suffix	Stem
studies	-es	study
studying	-ing	study
niñas	-as	niñ
niñez	-ez	niñ

$\longrightarrow$ **Stemming**

Lemmatization $\swarrow$

Form	Morphological information	Lemma
studies	Third person, singular number, present tense of the verb study	study
studying	Gerund of the verb study	study
niñas	Feminine gender, plural number of the noun niño	niño
niñez	Singular number of the noun niñez	niñez

The First Step: the preparation

In our analysis, **we also discard the order** in which words occur in documents, i.e., we assume that documents are a **bag of words**, where order does not inform our analyses

Is it a problem?

For instance, the expressions '*We are against lowering taxes, and for tax increases*' and '*We are for lowering taxes, and against tax increases*' use the exact same words, even though the meaning is reversed

The First Step: the preparation

While it is easy to construct sample sentences where word order fundamentally changes the nature of the sentence, empirically these sentences are rare

As a result, a simple list of words, which we call **unigrams**, is often sufficient to convey the general meaning of a text

The First Step: the preparation

We can also *retain some word-order* by including **bigrams** (word pairs, for example to distinguish the “White House” from the color and the domicile) or any other (defined as sequences of n consecutive tokens to form not words but phrases)

In practice, for common tasks, n-grams do little to improve the performance of text analysis

The First Step: the preparation

The result of the *preprocessing steps* is that each document can be represented as a **vector that counts** the number of times each of the unique words occur in each document

This the bag-of-words approach!

Multiple document vectors are then put together in a **document-term matrix (or document-feature matrix)**, where each **row** represents a document and each **column** represents a unique word, or term

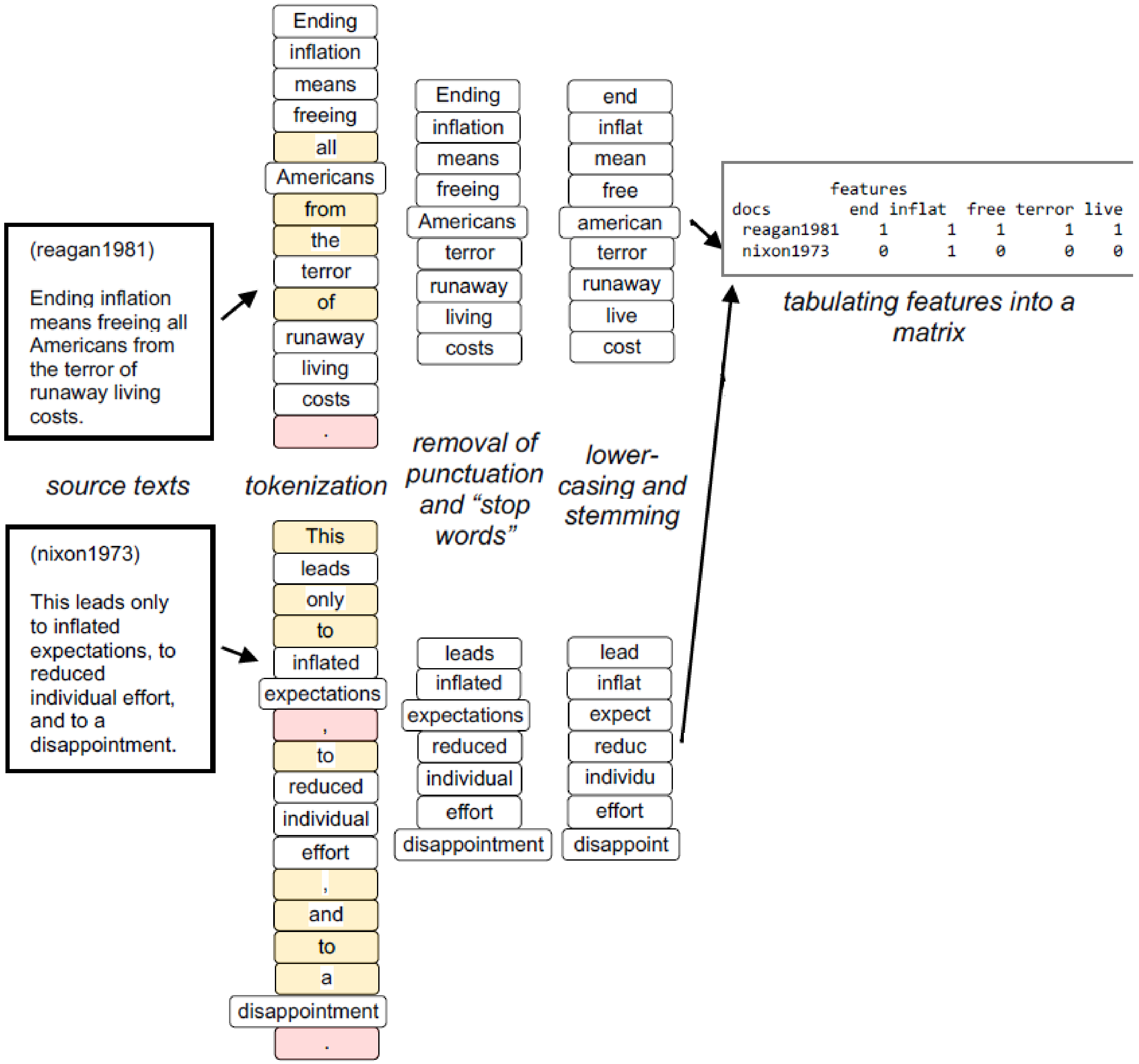
The First Step: the preparation

The matching between row and column will report either the frequency of that word in that document (as discussed above)….

….or alternatively a list of 0/1: where 0 if that word is not present in that document and 1 viceversa

This latter procedure is called **one-hot-encoding**

We will mainly deal with the former procedure - but not only: for example, a one-hot-encoding could be advisable given very short texts (such as tweets)



(reagan1981)

Ending inflation means freeing all Americans from the terror of runaway living costs.

(nixon1973)

This leads only to inflated expectations, to reduced individual effort, and to a disappointment.

```
            features
docs         end  inflat  free  terror  live
  reagan1981   1       1     1       1     1
  nixon1973    0       1     0       0     0
```

The First Step: the preparation

This matrix form of textual data can then be used as input into a variety of **analytical methods** for describing the texts

Quantitative text analysis thus moves textual data into the same domain as other types of quantitative data analysis, making it possible to bring to bear well-tested statistical and machine learning tools of analysis and prediction

The First Step: the preparation

Ironically, generating insight from text as data is only possible **once we have destroyed** our ability to make sense of the texts directly

To make it useful as data, we had to **obliterate the structure of the original text and turn its stylised and oversimplified features** into a glorified spreadsheet that no reader can interpret directly, no matter how expert in linear algebra you are!

The First Step: the preparation

We should not lose any sleep over it, because the point in analysing text as data was **never to interpret the data but rather to mine it for patterns**

(text) Mining is a destructive process - just ask any mountain! - and some destruction is inevitable in order to extract its valuable resources

And consistently across applications, scholars have shown that a simple representation of text such as the one we get via a bag-of-words approach is sufficient to infer **substantively interesting properties of texts**!

The First Step: the preparation

DfMs are however affected by what is known in machine learning as the **curse of dimensionality**: new observations tend to grow the feature set, and *each new term found in even one single document adds a new column to the matrix*

This usually creates a **problem of sparsity** in your dfm (a matrix with lots of 0s!) – often a statistical challange!

$$\begin{bmatrix} 0 & 1 & 5 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 7 & 0 & 9 \\ 0 & 0 & 0 & 4 & 0 \\ 0 & 2 & 0 & 0 & 8 \end{bmatrix}$$

The First Step: the preparation

Several of the pre-processing techniques just discussed allows to minimize precisely the sparsity problems

One further strategy for mitigating the problem of exponentially increasing dimensionality is to **trim** the document-feature(term) matrix

Trimming can be done on various criteria, but usually takes the form of a filter based on some form of feature frequency (i.e., keeping only features that appear just in 10% of documents for example)

The First Step: the preparation

Under some given circumstances, you could also prefer to **weight** your document-feature(term) matrix

Weighting schemes convert a matrix of counts into a matrix of weights

The most common of these is **relative term frequency**, a weighting process also known as document *normalisation* because it homogenises the sum of the counts for each document

Since documents in a typical corpus vary in length, this provides a method for comparing frequencies more directly than counts, which are inflated in longer documents

The First Step: the preparation

Words may also be weighted according to how *rare or frequent* they are in the corpus via a ***tf-idf* (term frequency-inverse document frequency)** matrix

tf-idf is a method in information retrieval for down-weighting the terms that are common to documents

tf-idf adds a weight that approaches zero as the number of documents in which a term appears (in any frequency) approaches the number of documents in the collection. When we have selected our texts because they pertain to a specific topic - as we usually will - then inverse document frequency weighting means **zeroing out most of our topical words**, since these will appear in most or all documents

The First Step: the preparation

In texts of debates over **health care**, for instance, tf-idf weighting is likely to eliminate all words related to health care, even when they might occur at very different rates across different documents

The First Step: the preparation

Note that many models commonly used in political science - such as the Wordfish model or topic models that we will see later on - only work with counts as inputs, so that tf-idf or other weighting schemes are inapplicable (but trimming yes!)

The First Step: the preparation

Never underestimate the *power* of the preprocessing stage!

Preprocessing has tremendous consequences for the quality of automated text analysis

The First Step: the preparation

Denny and Spirling (2018) replicated several published text analyses from political science using a variety of alternative feature processing steps

Their results shows that “under relatively small perturbations of of preprocessing decisions...very different substantive interpretations would emerge”

Researchers in practice should be aware of these decisions, critically examine the assumptions of their methods and how these relate to feature selection, and test the robustness of these results

Statistical summaries

Once you have your DfM, you can start to play! And tomorrow, we will begin to do that!

Before our first Lab class

If you **have a laptop** with you:

1) Install the latest version of R
2) For **Windows platforms**: install the latest version of Rtools (i.e., Rtools 4) from here (https://cran.r-project.org/bin/windows/Rtools/)
3) For **OS X**, do the following:

a) First try to install Quanteda directly

b) If you fail in doing that, install XCode from the App Store

Before our first Lab class

c) To install XCode, follow these simple rules:

1 Access to “Apple Developer”

https://developer.apple.com/download/more/

(You need Apple ID and password)

2 Insert “Xcode” in “Search Downloads” located on the left side of the page.

3 Choose “Xcode 12” and download.

4 After finishing download, click "Finder" and then "download." Double click “Xcode 12”. It may take a while to open this file

d) If you have problems to install the **latest version** of Xcode, **uses an earlier one**, such as Xcode 9!

Before our first Lab class

e) To make things even more complicated for Mac users: the latest R could not be compatible with the most recent Xcode. In that case, they the second most recent version of R

Before our first Lab class

Install the following packages by running these lines:

install.packages('devtools', repos='http://cran.us.r-project.org')

install.packages('quanteda', repos='http://cran.us.r-project.org')

install.packages('quanteda.textstats', repos='http://cran.us.r-project.org')

install.packages('readtext', repos='http://cran.us.r-project.org')

install.packages('ggplot2', repos='http://cran.us.r-project.org')

devtools::install_github("quanteda/quanteda.corpora")